基于节点多样性的域内路由保护算法 *

张 伟 [1]，耿海军 [2]

(1. 中国劳动关系学院 计算机应用教研室，北京 100048; 2. 山西大学 软件学院，太原 030006)

摘 要：已有的路由保护方案都没有考虑网络中节点的重要程度，然而在实际网络中不同节点在网络中的重要程度是不相同的。针对该问题，提出一种基于节点多样性的域内路由保护算法（intra-domain routing protection algorithm based on node diversity，RPBND）。首先，计算节点构造以目的为根的最短路径树（shortest path tree，SPT），从而保证 RPBND 算法和目前互联网部署的路由算法的兼容性；然后在该最短路径树的基础上构造特定结构的有向无环图（directed acyclic graph，DAG），从而最大化路由可用性。实验结果表明，RPBND 极大地提高了路由可用性，降低了故障造成的网络中断时间，从而为 ISPs 部署域内路由保护方案提供了充分的依据。

关键词：域内路由；路由保护；节点多样性；路由可用性

中图分类号：TP393.02 doi: 10.19734/j.issn.1001-3695.2018.06.0456

Intra-domain routing protection algorithm based on node diversity

Zhang Wei[1], Geng Haijun[2]

(1. *Computer Application Teaching & Research Section*, *China University of Labor Relations*, *Beijing* 100048, *China*; 2. *School of Software Engineering Shanxi University*, *Taiyuan* 030006, *China*)

Abstract: The existing routing protection schemes do not consider the importance of nodes in the network; however, the importance of different nodes in the network is not the same in real networks. To solve this problem, this paper proposed an intra-domain routing protection algorithm based on node diversity (RPBND) . Firstly, the computing node constructed a shortest path tree rooted at the destination node, which ensured the compatibility between the RPBND and the current deployment intra-domain routing algorithm. Then, it builded a directed acyclic graph on the basis of the shortest path tree, which maximized the Internet routing availability. The experiment results show that RPBND greatly improves the Internet routing availability and reduces the network disruption time caused by the faults, which provides the sufficient basis for ISPs to deploy the intra-domain routing protection scheme.

Key words: intra-domain routing; routing protection; node diversity; Internet routing availability

0 引言

随着互联网的快速发展和规模的急剧膨胀[1-3]，其应用已经深入到人们的学习、工作和生活等各个环节，在人们的日常生活中占据了重要的位置，并且已经成为全球最大的通信平台。随着互联网的发展，其应用范围也呈现出了显著的变化，互联网在设计之初主要支持一些非实时应用，如传送文件，发送电子邮件，聊天等应用。然而，现在大量的实时业务和关键业务[4]，如 VoIP、视频、在线手术、网上银行和股票交易等业务，在互联网上广泛传播，这些实时业务和关键业务对互联网的路由可用性提出了更加苛刻的要求[5,6]。

相关研究表明，网络中的故障频繁发生[7]。当网络出现故障时，目前互联网部署的域内路由协议，如开放最短路径优先（open shortest path first ,OSPF）[8]和中间系统到中间系统（intermediate system-to-intermediate system, IS-IS）[9]，需要经历几秒甚至几十秒完成收敛，在此过程中，将会出现路由环路和路由黑洞，从而导致网络中断、报文丢失[10]。互联网部署的动态路由协议难以满足实时业务和关键业务对网络时延的要求，因此学术界和工业界提出利用路由保护方案[11-13]来应对网络中频繁出现的故障。

目前互联网部署的域内路由保护方案有等价多路径（equal cost multiple paths ,ECMP）[8]和 LFA（loop free alternates）[14]。本文对这两种路由保护方案进行了深入的研究和分析，发现这两种方案有两个共同的特征。首先，基于目的地址的逐跳方式转发报文，没有借助辅助机制，如 Not-Via[15]，隧道（tunnel）[16]和多协议标签交换（multi-protocol label switching，MPLS）

收稿日期：2018-06-01；**修回日期：**2018-07-18 **基金项目：**国家自然科学基金资助项目（61702315）

作者简介：张伟（1982-），男，讲师，硕士，主要研究方向为计算机网络等；耿海军（1983-），男（通信作者），讲师，博士，主要研究方向为网络体系结构和路由算法等（ghj123025449@163.com）.

[17]等技术来实现报文的转发；其次，对于任意目的，报文的转发路径不会出现路由环路。基于目的地址的逐跳方式转发报文和目前互联网部署的路由协议的转发方式是相同的，因此 ECMP 和 LFA 支持增量部署，更利于实际操作。这两种方案的转发路径没有路由环路的是因为对于任意目的地址，所有节点的转发路径构成了一棵以目的为根的有向无环图。然而，利用 ECMP 和 LFA 方案构造的 DAG 并没有包含网络中所有的链路，有一些链路没有被充分利用，因此故障保护率较低。相关研究表明 LFA 的故障保护率仅仅为 50%[18]，为了进一步提高路由可用性，业界提出了 MARA-MC 方案[19]，该方案充分利用了网络中所有的链路，然而该方案并没有考虑节点的多样性。

已有的构造 DAG 的方案[20-2]都是为所有节点计算尽可能多的下一跳，从而提升路由可用性，然而所有的方案在计算 DAG 的时候都没有考虑网络中节点的多样性，然而在实际网络中不同节点在网络中的重要程度是不相同的。因此，本文提出一种基于节点多样性的域内路由保护算法 RPBND，从而最大限度的提高路由可用性，大大降低由于故障造成的网络中断时间。

1　网络模型和问题描述

1.1　网络模型

将网络表示为无向图 $G=(V,E)$，其中 V 表示网络中节点（路由器）的集合，E 表示网络中边（链路）的集合。对于网络 G 中的任意一条链路 $(i,j)\in E$，用 $w(i,j)$ 表示该链路的代价，$r(i,j)$ 表示该链路 $(i,j)\in E$ 的失效概率。对于网络 G 中的任意一个节点 $v\in V$，$Neighbor(v)$ 表示节点 v 的邻居节点的集合，$spt(v)$ 表示以节点 v 为根的最短路径树，$dag(v)$ 表示以节点 v 为根的有向无环图，$N(v,d)$ 表示节点 v 到节点 d 的下一跳集合。因为对于所有目的地址的计算方法是相同的，所以本文只讨论目的节点为 d 的算法。因此，为了描述简洁，用 $N(v)$ 表示 $N(v,d)$。

节点多样性是指对于任意的节点 $v\in V$，节点 V 的多样性表示该节点将报文成功转发到下一跳的概率，用 $A(v,S)$ 来表示，具体可以用公式表示为 $A(v,S)=1-\prod_{u\in N(v)} r(v,u)$，其中 S 表示节点的集合，将在 2.1 节详细介绍其具体内容。

下面通过一个例子来说明上述定义。图 1 表示以 d 为根的最短路径树，其中实线表示该树上的链路，虚线表示未使用的链路，边上的数值表示该边的失效概率。节点 a 和 b 到节点 d 的下一跳集合可以表示为 $N(a)=N(b)=d$。节点 a 的多样性可以

表示为 $A(a,S)=1-r(a,d)=0.9$，节点 b 的多样性可以表示为 $A(b,S)=1-r(b,d)=0.99$。图 2 表示以 d 为根的有向无环图，节点 a 到节点 d 的下一跳集合可以表示为 $N(a)=\{b,d\}$，节点 b 到节点 d 的下一跳集合可以表示为 $N(b)=\{d\}$。节点 a 的多样性可以表示为 $A(a,S)=1-r(a,d)*r(a,b)=0.99$，节点 b 的多样性可以表示为 $A(b,S)=1-r(b,d)=0.99$，从该例子可以看出，增加节点到目的下一跳数量可以提高节点的多样性。

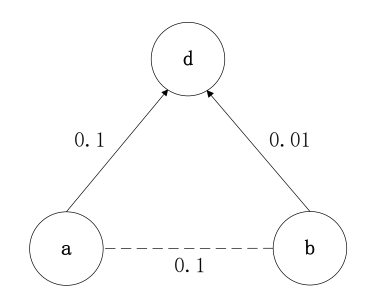


图 1　以 d 为根的最短路径树

Fig.1　the shortest path tree with d as root

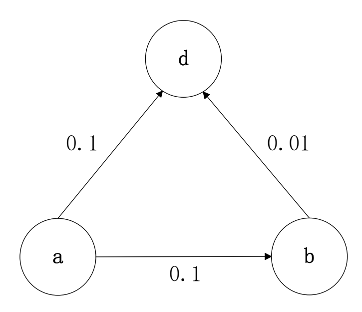


图 2　以 d 为根的有向无环图

Fig.2　directed acyclic graph with d as root

1.2　问题描述

从上述例子可以看出，构造以目的为根的 DAG 可以增加节点的多样性，从而提高路由可用性。然而，给定一个网络拓扑结构和目的节点，可以构造多个以目的为根的 DAG。为了实现路由可用性最大化的目标，本文需要构造一棵特定结构的 DAG，该 DAG 必须满足下面两个条件：

a）以目的为根的 DAG 包括以该目的为根的 SPT，从而确保和目前互联网部署的路由协议的兼容性。

b）最大化最小节点的多样性。

该问题可以形式化表示为

输入：网络拓扑结构 $G(V,E)$ 和目的节点 $d\in V$

输出：以 d 为根的有向无环图 $dag(d)$

目标：$\max \min_{u\in V} A(u,S)$

条件：$dag(d)\supseteq spt(d)$

2　集中式算法

2.1　算法描述

本文利用构造以目的为根的 DAG 来增加网络中节点的多样性，为了保证和互联网部署的域内路由协议的兼容性，该 DAG 需要包含 SPT，为了保证最大化节点的多样性，该 DAG 需要包含网络中所有的链路。因此，下面需要解决两个方面的问题：a）如何保证构造的有向无环图包含最短路径树；b）如何根据链路的质量确定链路在 DAG 中的方向。

问题 a）　首先构造一棵以 d 为根的最短路径树 $spt(d)$，然后在这棵树的基础上构造 $dag(d)$，这样就可以保证最终构造

的有向无环图包含最短路径树。

问题 b） 由于不在 $spt(d)$ 的每条链路都有两个方向，因此需要设定一个度量来确定上述链路在 $dag(d)$ 的方向，本文采用节点的多样性来解决该问题。

算法 1 描述了如何构造以 d 为根的有向无环图。算法的输入是网络拓扑结构 $G(V,E)$ 和目的节点 $d\in V$，输出是在 $spt(d)$ 上增加的链路的集合。首先初始化集合 S，M 和 C，其中 C 表示增加的链路的集合（第 1-3 行），根据网络中链路代价构造以 d 为根的最短路径树（第 4 行）。计算集合 M（第 6-10 行），该集合中的节点与集合 S 中的节点有直接相连的边，并且该边在 $spt(d)$ 中，从而保证有向无环图包含 $spt(d)$。从集合 M 中选择具有最大多样性的节点加入到集合 S 中，更新集合 S 和 M（第 11-13 行）。对于网络中所有不属于 $spt(d)$ 的边 (u,v)，如果节点 v 比节点 u 先加入到集合 S 中，则将边 (u,v) 加入到集合 C 中（第 15-19 行），最后返回集合 C（第 20 行）。

Algorithm 1

Input: $G(V,E)$，目的地址 d。

Output: 需要添加的边的集合 C。

1： $S\leftarrow\{d\}$

2： $M\leftarrow\emptyset$

3： $C\leftarrow\emptyset$

4：构造以 d 为根的最短路径树 $spt(d)$

5： **while** $|S|<|V|$

6： **for** $u\in V-S\ \&\&\ v\in S$ **do**

7： **if** $(u,v)\in spt(d)$ **then**

8： $M\leftarrow\{u\}\cup M$

9： endif

10： endfor

11： $w\leftarrow argmax_{w\in M}(A(w),S)$

12： $S\leftarrow\{w\}\cup S$

13： $M\leftarrow\emptyset$

14：endwhile

15： **for** $(u,v)\in E$ && $(u,v)\notin spt(d)$ **do**

16： **if** $Location(S,v)<Location(S,u)$ **then**

17： $C\leftarrow C\cup(u,v)$

18： **endif**

19：endfor

20： **return** C

2.2 算法举例

下面通过一个例子来说明算法 1 的执行过程。初始化：$S\leftarrow\{d\}$，$M\leftarrow\emptyset$和 $C\leftarrow\emptyset$，构造以目的地址 d 为根的最短路径树（见图 1，假设链路代价为跳数），节点 a 到 d 的下一跳为 d，b 到 d 的下一跳为 d。

a）第一次循环。更新集合 $M\leftarrow\{a,b\}$，根据节点多样性公式计算 $A(a,S)$ 和 $A(b,S)$，$A(a,S)=0.9$，$A(b,S)=0.99$，因此 $w\leftarrow b$，更新集合 S，$M\leftarrow\emptyset$。

b）第二次循环。更新集合 $M\leftarrow\{a\}$，$A(a,S)=0.99$，因此 $w\leftarrow a$，更新集合 S，$M\leftarrow\emptyset$。

最后根据集合 S 计算需要增加的边的集合为 $C\leftarrow\{(a,b)\}$。

假设在算法执行过程中不考虑节点多样性的数值，随机将节点加入到集合 S 中，则可能导致节点多样性较低，下面通过例子来说明这种情况。如果在第一次循环中选择节点 a 加入到集合 S，则 $A(a,S)=0.9$，在第二次循环中选择节点 b 加入到集合 S，则 $A(b,S)=0.999$，网络中节点多样性的最小值为 $A(a,S)=0.9$。因为本文算法的目标是最大化最小节点多样性，因此这种算法显然不是最优的解决方案。

2.3 算法时间复杂度

定理 1 算法 1 的时间复杂度为 $O(|V|lg|V|+|E|)$。

证明 算法需要构造一棵以 d 为根的最短路径树（第 4 行），该算法的时间复杂度为 $O(|V|lg|V|+|E|)$，更新集合 M 的时间复杂度为 $O(|V|lg|V|+|E|)$（第 6-10 行）。从集合 M 中选择具有最大节点多样性的节点的时间复杂度为 $O(lg|V|)$（第 11 行），执行 $|V|$ 次的时间复杂度为 $O(|V|lg|V|)$（第 5 行）。第 15-19 行的时间复杂度为 $O(|E|)$。因此算法 1 的时间复杂度为 $O(|V|lg|V|+|E|)$。

2.4 算法正确性

定理 2 对于网络 G 中的任意一个节点 $v\in V$，如果集合 $B\subset C$，则 $A(v,B)\le A(v,C)$。

证明 当集合为 B 时，假设节点 v 到目的的下一跳集合为 N_B，当集合为 C 时，假设节点 v 到目的的下一跳集合为 N_C，因为 $B\subset C$，所以 $N_B\subseteq N_C$。根据节点多样性的定义可得 $A(v,B)\le A(v,C)$。

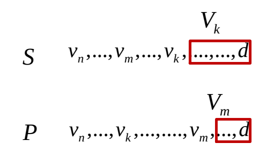


图 3 定理 2 的证明图示

Fig.3 Illustration of the proof of theorem 2

定理 3 利用算法 1 可以计算出集合 S 的最优解。

证明 下面利用反证法来证明该定理。

利用算法 1 计算出的集合 $S=(v_n,v_{n-1},\ldots v_{k+1},v_k\ldots d)$，其中 $n=|V|$。假设 v_k 为最先加入到集合 S 中，并且具有最小节点多样性的节点。假设存在一个节点 v_m 并且 $v_m\in(v_n,v_{n-1},\ldots v_{k+1})\subset S$，在集合 S 中节点 v_m 在节点 v_k 之后加入到集合 S 中，而在集合 $P=(v_n,v_{n-1},\ldots,v_k\ldots v_m\ldots d)$ 中，该节点比节点 v_k 先加入到集合 P 中，并且 $A(v_m,P)>A(v_k,S)$。令 V_m 表示比节点 v_m 先加入到集合 P 中的节点的集合，V_k 表示比节点 v_k 先加入到集合 S 中的节点的集合，如图 3 所示。因为 $V_m\subset V_k$，根据定理 2 可知 $A(v_m,V_m)\le A(v_m,V_k)$（1）。在集合 S 中，因为节点 v_k 比节点 v_m 先加入到集合 S 中，所以 $A(v_m,V_k)\le A(v_k,V_k)$（2）。根据（1）和（2）可知 $A(v_m,V_m)\le A(v_k,V_k)$，即 $A(v_m,P)\le A(v_k,S)$ 这与前提假设 $A(v_m,P)>A(v_k,S)$ 相矛盾，因此定理得证。

3　分布式算法

在上节详细描述了集中式算法，该方案可以利用 SDN[22]技术进行部署。然而，为了在目前互联网部署的路由协议上部署该方案，需要研究如何利用分布式算法解决上述问题。在集中式算法中，计算节点需要为所有的节点计算下一跳集合，然而在分布式算法中，计算节点仅仅需要为自己计算下一跳集合即可。因此，与集中式算法比较，分布式算法可以大大降低算法的计算时间和存储空间。分布式算法只需要对集中式算法做微小的调整，即当某个节点运行算法时，如果该节点或者该节点的所有邻居加入到集合 S 中，则算法执行完毕。

下面详细描述分布式算法的执行过程。对于任意目的地址 d，如果运行算法的节点为 n，只需要对算法 1 作如下修改：

a）将 Input 修改为 $G(V,E)$，目的地址 d，运行算法节点 n

b）将第 5 行 While 条件修改为 $|S|<|V|\&\&!(n\in S||Neighbor(n)\subseteq S)$,表示节点 n 或者其所有的邻居加入到集合 S 中,算法将执行完毕。

c）将第 15~19 行中所有的变量 u 修改为变量 n，表示算法只需要考虑与节点 n 直接相连的边即可。

4　实验

4.1　实验方法

1）实验拓扑

为了充分准确的评价算法的性能，在实验中采用了大量的拓扑结构，其中包括，真实拓扑结构 Abilene[2]，Rocketfuel[23]项目公开的测量拓扑结构，利用 Brite 软件[2]产生的拓扑。

a）Abilene 是美国用于科研的网络，该拓扑包括 11 个节点和 14 条边。

b）在 Rocketfuel 项目公开的拓扑结构中，选取了 6 个作为本文的实验拓扑结构，见表 1.

c）Brite 是一个产生拓扑的开源软件，参数见表 2。

表 1　Rocketfuel 拓扑结构

Table 1　Rocketfuel topological structure

AS 号码	AS 名称	节点数量	链路数量
1221	Telstra	108	153
1239	Sprint	315	972
1755	Ebone	87	162
3257	Tiscali	161	328
3967	Exodus	79	147
6461	Abovenet	128	372

表 2　Brite 生成拓扑结构的参数设置

Table 2　Parameter settings for topology by Brite

模型	节点数量	HS	LS
Waxman	1000	1000	100
链路节点比	NodePlacement	增长方式	alpha
2-10	Random	增量式	0.15
beta	BWDist	BwMin-BwMax	模式
0.2	Constant	10.0-1024.0	路由器

2）评价指标

本文将 RPBND 和 OSPF、LFA、MARA-MC 进行比较，主要比较这几种算法的计算复杂度和路由可用性。本实验的模拟环境是 PC 机， CPU 是 Intel i7，主频 1.7GHz，内存 2 GB，采用 C++开发模拟平台，实验结果为 50 次计算结果的平均值。

4.2　计算复杂度

本节讨论四种算法的计算复杂度，包括每个节点的计算复杂度和所有节点的计算复杂度。表 3 列出了四种算法的计算复杂度，从表 3 可以看出，运行一次 Dijkstra 的算法的计算复杂度为 $O(VlgV+E)$，网络中所有节点运行 Dijkstra 的算法的计算复杂度为 $V*O(VlgV+E)$。为了实现 LFA，每个节点需要运行多次 Dijkstra，计算复杂度为 $D*O(VlgV+E)$，其中 D 表示计算节点的度，网络中所有节点运行 LFA 的时间复杂度为 $V*O(VlgV+E)$。MARA-MC 和 RPBND 的计算复杂度是相同的，都是 $V*O(VlgV+E)$。

表 3　不同算法对应的计算复杂度

Table 3　Computational complexity of different algorithms

算法	分布式算法	集中式算法
Dijkstra	$O(\|V\|lg\|V\|+\|E\|)$	$O(\|V\|lg\|V\|+\|E\|)$
LFA	$\|D\|*O(\|V\|lg\|V\|+\|E\|)$	$\|V\|*O(\|V\|lg\|V\|+\|E\|)$
MARA-MC	$\|V\|*O(\|V\|lg\|V\|+\|E\|)$	$\|V\|*O(\|V\|lg\|V\|+\|E\|)$
RPBND	$\|V\|*O(\|V\|lg\|V\|+\|E\|)$	$\|V\|*O(\|V\|lg\|V\|+\|E\|)$

4.3　路由可用性

本节将利用路由可用性来衡量不同算法的性能。假设源节点到 s 目的节点 d 有 k 条路径，$l_j(s,d)$ 表示其中的第 j 条路径，该路径的可用性可以表示为 $B(l_j(s,d))=\prod_{(m,n)\in l_j(s,d)}1-r(m,n)$，则节点 s 到节点 d 的端到端的可用性可以表示为 $F(s,d)=\sum_{i=1}^{k}(-1)^{i-1}M_i$，其中 $M_i=\sum_{e<f<\ldots g}B(l_e(s,d))\cap B(l_f(s,d))\ldots\cap B(l_g(s,d))$，可以表示为 $M_i=\sum_{e<f<\ldots g}\prod_{(m,n)\in(l_e(s,d)\cup l_f(s,d)\ldots\cup l_g(s,d))}1-r(m,n)$，因此路由可用性可以表示为 $A(G)=\frac{\sum_{s,d\in V}F(s,d)}{|V|*(|V|-1)}$。为了简化实验，本文采用一种简单的模型模拟网络中链路的失效概率，假设网路中链路的失效概率为 0 到 0.05 之间的随机数。

首先，描述不同算法在真实拓扑和测量拓扑的运行结果。表 4 列出了不同算法对应的路由可用性。从该表可以看出，RPBND 算法在所有拓扑结构中的路由可用性的性能都明显优于其他三种算法，并且 RPBND 算法的路由可用性在所有拓扑上的运行结果都在 97%以上。

接着讨论不同算法在模拟拓扑上的运行结果。图 4 描述了当网络拓扑大小为 1000，路由可用性和网络节点平均度的关系。从图 4 可以看出，随着网络节点平均度的增加，除 Dijkstra 外其余算法的路由可用性也随之增加，这是因为随着节点度的增

加，网络中路径的多样性将会增加，而 Dijkstra 始终只计算最优路径。RPBND 算法的路由可用性依然明显优于其他所有算法。

表 4 算法在真实拓扑和测量拓扑中的路由可用性

Table 4 Routing availability in real topology and measurement topology

AS 名称	Dijkstra	LFA	MARA-MC	RPBND
Telstra	89.34%	92.45%	94.67%	98.45%
Sprint	83.45%	90.35%	93.26%	97.56%
Ebone	79.34%	91.24%	94.56%	98.67%
Tiscali	91.34%	93.67%	95.68%	98.87%
Exodus	80.34%	90.23%	93.67%	98.54%
Abovenet	83.47%	89.67%	92.56%	97.68%

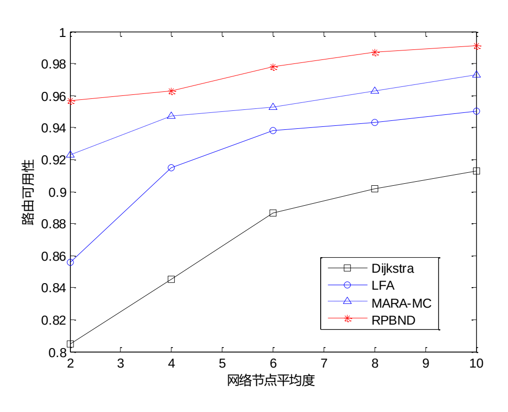


图 4 网络节点平均度和路由可用性关系

Fig.4 Relationships of network node average and routing availability

5 结束语

为了满足实时应用对路由可用性的要求，业界提出利用路由保护来解决该问题。然而，已有的算法都没有考虑网络中节点多样性，因此，已有的算法并不能计算出最优路由。本文提出了一种基于节点多样性的路由保护算法 RPBND，为了保证和互联网部署的域内路由协议的兼容性，该算法首先构造一棵以计算节点为根的最短路径树，然后根据节点的多样性，基于上述的最短路径树构造有向无环图，从而最大化路由可用性。

链路失效概率直接关系到节点多样性的计算。因此下一步将重点研究链路失效模型，利用该模型更加准确的进行实验。下一步将利用虚拟化技术将该算法部署在 CERNET2[26]上，利用实际测试结果进一步优化设计方案。

参考文献：

[1] Geng Haijun, Shi Xinggang, Yin Xia, *et al.* Algebra and algorithms for multipath QoS routing in link state networks [J]. Journal of Communications and Networks, 2017, 19 (2): 189-200.

[2] Geng Haijun, Shi Xingang, Wang Zhiliang *et al.* A hop-by-hop dynamic distributed multipath routing mechanism for link state network [J]. Computer Communications, 2018, 116: 225-239.

[3] Gopalan A, Ramasubramanian S. IP fast rerouting and disjoint multipath routing with three edge-independent spanning Trees [J]. IEEE/ACM Trans on Networking, 2016, 24 (3): 1336-1349.

[4] Antonakopoulos S, Bejerano Y, Koppol P. Full protection made easy: the DisPath IP fast reroute scheme [J]. IEEE/ACM Trans on Networking, 2015, 23 (4): 1229-1242.

[5] Yang Baohua, Liu Junda, Scott Shenker *et al.* Keep forwarding: Towards k-link failure resilient routing [C]// Proc of the IEEE International Conference on Computer Communications. Piscataway, NJ: IEEE Press, 2014: 1617-1625.

[6] Zheng Jiaqi, Xu Hong, Zhu Xiaojun, *et al.* We've got you covered: failure recovery with backup tunnels in traffic engineering [C]// Proc of the 24rd IEEE International Conference on Network Protocols. Piscataway, NJ: IEEE Press, 2016: 1-10.

[7] Markopoulou A, Iannaccone G, Bhattacharyya S. Characterization of failures in an operational ip backbone network [J]. IEEE/ACM Trans on Networking, 2008, 16 (4): 749-762.

[8] Moy J. RFC 2328, OSPF version 2 [S]. 1998.

[9] Oran D. RFC 1142, OSI IS-IS intra-domain routing protocol [S]. 1990.

[10] Yallouz J, Rottenstreich O, Babarczi P, *et al.* Optimal link-disjoint node-"somewhat disjoint"paths [C]// Proc of the 24th IEEE International Conference on Network Protocols, Piscataway, NJ: IEEE Press, 2016: 1-10.

[11] Xu Mingwei, Hou Meijia, Wang Dan, *et al.* An efficient critical protection scheme for intra-domain routing using link characteristics [J]. Computer Networks, 2013, 57 (1): 117-133.

[12] Yang Yuan, Xu Mingwei, Li Qi. Fast rerouting against multi-link failures without topology constraint [J]. IEEE//ACM Trans on Networking, 2018 (99): 1-14.

[13] Lee S, Yu Yinzhe, Nelakuditi S, *et al.* Proactive vs reactive approaches to failure resilient routing [C]// Proc of IEEE International Conference on Computer Communications. Piscataway, NJ: IEEE Press, 2004: 1-9.

[14] Atlas A K, Zinin A. RFC 5286, Basic Specification for IP fast reroute: loop-free alternates [S]. 2008.

[15] Enyedi G, Rétvári G, Szilágyi P, *et al.* IP Fast reroute: lightweight not-via without additional addresses [C]// Proc of IEEE International Conference on Computer Communications, Piscataway, NJ: IEEE Press, 2009: 157–168.

[16] Xu Mingwei, Yang Yuan, Li Qi. Selecting shorter alternate paths for tunnel-based ip fast reroute in linear time [J]. Computer Networks, 2012, 56 (2): 845–857.

[17] Sommers J, Barford P, Eriksson B. On the prevalence and characteristics of MPLS deployments in the open Internet [C]// Proc of ACM SIGCOMM conference on Internet measurement conference. New York: ACM Press, 2011: 445-462.

[18] Rétvári G, Tapolcai J, Enyedi G, *et al.* IP fast reroute: loop free alternates revisited [C]// Proc of IEEE Conference on Computer Communications. Piscataway, NJ: IEEE Press, 2011: 2948-2956.

[19] Ohara Y, Imahori S, Meter R V. MARA: maximum alternative routing



algorithm [C]// Proc of IEEE Conference on Computer Communications. Piscataway, NJ: IEEE Press, 2009: 298-306.

[20] Kwong K W, Gao Lixin, Guérin, *et al.* On the feasibility and efficacy of protection routing in IP networks [C]// Proc of IEEE International Conference on Computer Communications. Piscataway, NJ: IEEE Press, 2010: 1235–1243.

[21] Cho S, Elhourani T, Ramasubramanian S. Independent directed acyclic graphs for resilient multipath routing [J]. IEEE/ACM Trans on Networking, 2012, 20 (1): 153–162.

[22] Nunes B, Mendonca M, Nguyen X N, *et al.* A survey of software-defined networking: past, present, and future of programmable networks [J]. IEEE Communications Surveys & Tutorials, 2014, 16 (3): 1617–1634.

[23] Spring N, Mahajan R, Wetherall D, *et al.* Measuring ISP topologies with rocketfuel [J]. IEEE//ACM Trans on Networking, 2004, 12 (1): 2 – 16.